

B O L T B E R A N E K A N D N E W M A N I N C
C O N S U I T I N G • D E V E L O P M E N T • R E S E A R C H

Report No. 3106

✓ INTERFACE MESSAGE PROCESSORS FOR
THE ARPA COMPUTER NETWORK

(QUARTERLY TECHNICAL REPORT No. 2)

(1 April 1975 to 30 June 1975)

Principal Investigator: Mr. Frank E. Heart
Telephone (617) 491-1850, Ext. 470

Sponsored by:
Advanced Research Projects Agency
ARPA Order No. 2351, Amendment 15
Program Element Codes 62301E, 62706E, 62708E

✓ Contract No. F08606-75-C-0032
Effective Date: 1 January 1975
Expiration Date: 31 December 1975
Contract Amount: \$2,384,745

D D C
RECEIVED
AUG 11 1975
B

Title of Work: Operation and Maintenance of the ARPANET

Submitted to:

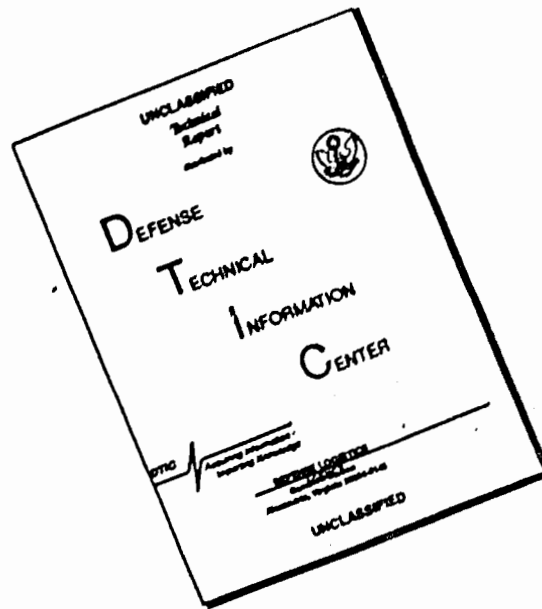
IMP Program Manager
Range Measurements Lab.
Building 981
Patrick Air Force Base
Cocoa Beach, Florida 32925

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.




AD A013370

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

ACCESSION for	
NTIS	White Section <input checked="" type="checkbox"/>
DGC	Buff Section <input type="checkbox"/>
UNANNOUNCED	<input type="checkbox"/>
JUDICIALIZATION	
BY _____	
DISTRIBUTION TO DEPARTMENT OF JUSTICE	
Dist.	APRIL 6 1964
	

11 Jul 75

1-83F.

6 INTERFACE MESSAGE PROCESSORS FOR
THE ARPA COMPUTER NETWORK.

10 Frank L. Herit

15 F08606-75-C-0032,
✓ VARP Order-2351

9 QUARTERLY TECHNICAL REPORT NO. 2,
1 Apr 75 - 3 Jun 75.

Submitted to:

IMP Program Manager
Range Measurements Lab
Building 981
Patrick Air Force Base
Cocoa Beach, Florida 32925

This research was supported by the Advanced Research Projects
Agency of the Department of Defense and monitored by the Range
Measurements Laboratory under Contract No. F08606-75-C-0032.

060 100

mt

TABLE OF CONTENTS

	Page
1. OVERVIEW	1
2. NETWORK PERFORMANCE, SOFTWARE, AND DEVELOPMENTS . . .	6
2.1 TIP Performance and Improvements	6
2.2 IMP Performance and Improvements	19
2.3 New Developments	36
3. TIP ACCESS CONTROL AND ACCOUNTING	51
3.1 Development of the RSEXEC TIPSER System	54
3.2 Current TIP/RSEXEC Capabilities	57
3.3 Fundamental Structures	64
3.4 Discussion	71

1. OVERVIEW

This Quarterly Technical Report, Number 2, describes aspects of our work on the ARPA Computer Network under Contract No. F08606-75-C-0032 during the second quarter of 1975. (Work performed in 1973 and 1974 under Contract No. F08606-75-C-0027 has been reported in an earlier series of Quarterly Technical Reports, numbered 1-8; and work performed in 1969 through 1972 under Contract No. DAHC-69-C-0179 has been reported in a still earlier series of Quarterly Technical Reports, numbered 1-16.)

Two new IMPs were delivered during the second quarter, one to New York University (NYU) and the other to the National Security Agency (NSA). As noted in our last Quarterly Technical Report, an IMP was previously delivered to NYU but, because of a major telephone company fire, this IMP was re-shipped to Stanford Medical Center during the first week of the second quarter. A new IMP was procured by ARPA and shipped to NYU in time to meet the availability of communications circuits in mid-May. The second new IMP, which we shipped to NSA during the second quarter, is scheduled for connection to the network early in the third quarter.

During the second quarter ARPA decided that one of the two large Pluribus IMPs which we are constructing should be installed

at the Seismic Data Analysis Center (SDAC) during the third quarter. The large Pluribus IMPs under construction were not previously earmarked for delivery to any particular location, and hence were not configured to meet any particular site's requirements. Thus, a considerable amount of work remains to be done in order to configure one of these IMPs for the requirements of SDAC, where a total of five modem interfaces and four Host interfaces are required. Some of these interfaces are available, either because they were included in the original configuration of these machines or because they can be borrowed from other Pluribus machines. The remainder are now under construction.

The Pluribus Satellite IMP project continued to advance during the second quarter. In particular, by the end of the quarter a Pluribus Satellite IMP was able to communicate successfully with two 316 Satellite IMPs over our satellite channel simulator.

Work on the PLIs was concentrated on preparation for TEMPEST testing which is scheduled to begin early in July, as described in our previous Quarterly Technical Report. By the end of the quarter the shielded enclosure fabrication had been completed and one secure PLI had been repackaged in this enclosure. Shipment of a secure PLI, a bitstream PLI, and a 316 IMP to NESSEC for TEMPEST testing is expected to proceed on schedule.

In our last Quarterly Technical Report we mentioned that an environmental test chamber, to be used in our quality control program for new hardware, had been completed. During the second quarter nine Honeywell 316s (the majority of these were not for delivery to ARPA) and one Pluribus were run in this chamber at both high and low temperatures. Except for one 316, all experienced failures. These failures did not occur until temperatures greater than 85 degrees Fahrenheit were reached. In most cases, the failures remained in the machines even after temperature was returned to a normal setting of 70 degrees to 75 degrees Fahrenheit. This experience indicates that testing machines at elevated temperatures is to some extent an accelerated life test.

In the future we anticipate that all new machines will be subjected to two temperature tests. The first will take place several weeks before shipment, and consist of 24 hours of testing at 60 degrees Fahrenheit and 24 hours of testing at a higher temperature. For a 316, the high temperature test should be at 100 degrees Fahrenheit; some experiments are required to determine a suitable temperature for Pluribus equipment. The second test will be incorporated in the final quality control procedure and last 24 hours, divided equally between high and low temperatures. We expect that this testing prior to delivery will significantly reduce problems in the field.

During the second quarter we participated in a meeting of IFIP Working Group 6.1 which was primarily devoted to drafting recommendations to CCITT on various design aspects of packet-switching communications networks. With ARPA's consent we then participated in a CCITT Rapporteur's meeting on Question VII-1, Point C, as described in our last Quarterly Technical Report. The Rapporteur's group is attempting to develop draft recommendations for international public packet-switched services; our role in these deliberations was as technical experts serving the United States delegation.

During the second quarter we prepared a new operational document, BBN Report 2999, "Pluribus Document 1: Overview". In addition, several professional papers were presented, as follows: "Pluribus--A Reliable Multiprocessor," by S.M. Ornstein, W.R. Crowther, M.F. Kralej, R.D. Bressler, A. Michel, and F.E. Heart, and "Issues in Packet-Switching Design," by W.R. Crowther, F.E. Heart, A. A. McKenzie, J.M. McQuillan, and D.C. Walden, both presented at the AFIPS 1975 National Computer Conference, May 1975, Anaheim, California; "The Evolution of a High Performance Modular Packet-Switch," by S.M. Ornstein and D.C. Walden, and "A Dynamic Packet-Switching System for Satellite Broadcast Channels" by R. Binder, both presented at the 1975 International Conference on Communications, June 1975, San Francisco, California; "The

Pluribus Multiprocessor" by M.F. Kraley, presented at the 1975 International Symposium on Fault-Tolerant Computing, June 1975, Paris, France; and "Pluribus: A Multiprocessor for Communications Networks," by R.D. Bressler, M.F. Kraley, and A. Michel, presented at the Fourteenth Annual ACM/NBS Technical Symposium--Computing in the mid-70's: an Assessment, June 1975, Gaithersburg, Maryland. We also prepared and submitted two professional papers as follows: "Gateway Design for Computer Network Interconnection," by R.D. Rettberg and D.C. Walden, to be presented at the European Computing Conference on Communications Networks, September 23-25, 1975, in London, England; and "Techniques for Detecting and Preventing Interrupt Bugs," by B.P. Cosell, J.M. McQuillan, and D.C. Walden, to be presented at the IFIP/TC-2 Working Conference on Software for MiniComputers, September 8-12, 1975, at Lake Balaton, Hungary.

2. NETWORK PERFORMANCE, SOFTWARE, AND DEVELOPMENTS

In Section 8 of Quarterly Technical Report No. 1 in the current series, we discussed, in general terms, a study of network performance which we undertook during the first quarter and which continued well into the second quarter. In the rest of this section we discuss a number of specific results of the network performance study, first in the area of TIP performance and second in the area of IMP performance. Finally, we discuss a number of new developments.

2.1 TIP Performance and Improvements

As discussed in our previous Quarterly Technical Report, within a short time after release of Version 327 to all network TIPs, a major problem became evident. Most TIPs are equipped with 28 kilowords of core memory; of this 16K is dedicated to the IMP and the remainder to the TIP. The 12K TIP core must accommodate both the TIP code (which occupies the majority of the space) and terminal buffering. The new code needed for the access control and user accounting mechanisms reduced the amount of space available for terminal buffering (in a 28K TIP) to about two-thirds of that available with the preceding software version. Although this buffer reduction occurred in all TIPs, its effects

(frequency of the user typing fast enough to completely fill his input buffer and noticeable "stuttering" on output were most strongly felt at those TIPs supporting large numbers of terminals. Therefore, by the end of last quarter, a return to the previous TIP version (TIP Version 322) was made at all TIP sites with heavy terminal usage but only 12K of TIP memory. The remainder of the TIP sites continued to run TIP Version 327, but with the TIP access control and accounting mechanisms disabled as discussed in the previous report and in Section 3 of this report. Early in this quarter, TIP Version 327 was replaced by TIP Version 337 at all sites able to run it, i.e., those with 16K of TIP memory or low terminal usage. Near the end of this quarter, these same sites were converted to use of TIP Version 350. Both TIP Version 337 and 350 were mainly aimed at improved TIP performance (rather than simply at expanding the list of TIP features). Version 337 is described in the following subsection. Version 350, in addition to correcting a few minor bugs, includes the performance-improving mechanism described in Subsection 2.1.3 below. As additional memory has been added to those TIPs with only 12K of TIP memory, expanding them to the maximum of 16K of TIP memory, those TIPs have been converted from running TIP Version 322 to running TIP Version 350 (or its immediate predecessors, TIP Versions 337 or 327).

One correction to our statements about TIP performance as given in the previous Quarterly Technical Report is in order: in Section 2.6 of the previous report we noted that a long-standing protocol violation was found in the connection protocol implemented for the PDP-15 at ARPA. It has been pointed out to us that while a protocol violation in the PDP-15 was at one time suspected, the actual problem was eventually positively diagnosed to be a result of an intermittent hardware problem in the interface between the PDP-15 and its IMP.

2.1.1 TIP Version 337

TIP version 337 had little that was new. It was primarily a clean reassembly of version 327 including all the patches that were made to that version plus fixes to problems which were discovered in version 327 but were too difficult to patch. To enumerate the changes briefly: a) a suspended connection is cleared correctly in the case when its link gets re-used by the remote Host for some other connection; b) the TIP's "logger" will allow ICPs to distinct sockets on a single Host to proceed in parallel; c) the "logger" can abort ICPs more cleanly; d) suspended and restored TENEX connections are handled more cleanly; e) minor bugs in the user accounting and authentication

code were fixed; e) a number of miscellaneous minor bugs unrelated to network performance were fixed; f) routines were added to enable TIP tables to be sent through the network to the Network Control Center; and g) the copy-down loop in the TIP's RFNM logic was modified to allow the IMP (which co-resides with the TIP) to service I/O interrupts while the TIP is in its copy loop. The mechanism of point f has proved very useful for obtaining status and diagnostic information on the TIP. The change of point g was made to benefit the IMP.

2.1.2 Too Fast TIP Clock

One of the diagnostic mechanisms we have developed is a facility for real-time monitoring of IMP/TIP "load average". One of the results of this monitoring was the discovery that the Tymshare TIP when totally idle appeared busier than any other totally idle TIP in the network. We quite rapidly came to the hypothesis that one of the real time clocks (there are two, one for the IMP and one for the TIP) was interrupting more often than it was supposed to be. With a trivial program patch to count the frequency of the clocks it was found that the TIP clock was running at twice its nominal frequency, causing interrupts twice as often as it should, and therefore wasting lots of machine cycles running all the TIP code which runs at clock interrupt

tim. A field engineer had little problem correcting the hardware difficulty which caused the clock to run at double tempo. As a result of this discovery at Tymshare, all of the other IMP/TIP clocks in the network were also tested, and no other was found to be operating at the wrong frequency. Further, the Network Control Center has established a procedure for routine periodic testing of all the clocks in the network for correct frequency.

2.1.3 The Link 0 Blocking Phenomenon

Link 0 is used by the Host-to-Host protocol for all its control messages. In particular, ALLocate control messages must flow from the TIP to another Host if data is to flow from the other Host to the TIP (to be printed on a TIP terminal) on a data connection. A phenomenon we have observed is that when a number of terminals on a given TIP are simultaneously attempting to print output from the same other Host, terminal printing will be frequently interrupted for a second or a fraction of a second. This phenomenon has been traced to a problem with contention among the several terminals for link 0 between the TIP and the other Host: a terminal interrupts printing because the TIP has received no further output from the other Host; the Host has not sent further output because it has not received the necessary

ALLocate message from the TIP; the TIP has not sent the ALLocate because link 0 to the other Host is blocked waiting for a RFNM for a previously sent ALLocate; and Host-to-Host protocol currently prohibits a second ALLocate from being sent while the RFNM for a previous ALLocate is still outstanding. Thus, although the other Host is ready to send more output and the TIP is ready to print it, the entire system must wait for the outstanding RFNM to arrive, for the necessary ALLocate to be sent and find its way across the network to the other Host, and for the now-allowed data message to traverse its way back across the network to the TIP. We have developed an "extension" to the protocol which allows the TIP to send several ALLocate messages on link 0 without waiting for outstanding RFNMs, thus removing much of the latency from the ALLocate sending process and enabling smoother data output from the other Host to the TIP. (It should be noted that the TIP has always packed as many ALLocates into a single link 0 message as possible. The extension described here is to handle the case of a buffer becoming free just after a link 0 message was sent.) We have implemented this extension, and a version of the TIP software including this extension (TIP Version 350) is currently running at many TIP sites. We have been fortunate in discovering a mechanism which does not require any Hosts other than the TIP to make changes to their Network Control Programs.

2.1.4 Optional Removal of the 2741 Handler

A number of times in the past we have pointed out the possibility of removing the TIP's capability to handle IBM 2741 terminals (and their equivalents) on an optional basis. That is, for each TIP site a decision could be made whether the TIP should have the capability to handle 2741s or not. At sites where it was decided not to support the 2741 capability in the TIP, a significant amount of memory normally dedicated to 2741 handling could be used instead for terminal buffering. Although this option has been suggested several times in the past, until recently we have not managed to actually begin implementing it. However, while constructing the mechanism which leads to the solution of the link 0 blocking phenomenon discussed in Section 2.1.3 above, we figured out a relatively easy way to finally implement the 2741 removal option. In fact, the version of the TIP containing the mechanism to get around the link 0 blocking phenomenon is already a significant step toward the 2741 removal option, and changes we are planning to make to the TIP in other areas will naturally result in certain additional TIP structures being changed which will bring things very close to the point where removal of the 2741 capability is a real option.

2.1.5 TIP ALlocate and Buffering Strategy

During the TIP performance study, some questions arose regarding the performance of the TIP's strategy for buffering traffic arriving from another Host for printing on a TIP terminal and the TIP's strategy for sending the Host-to-Host protocol ALlocate messages which control the flow of data from the other Host to the TIP. In particular, some of the interested parties framed a very cogent presentation of their doubts about the TIP's strategy, and we reproduce this presentation in full below.

Our understanding of the TIP buffer allocation policy is as follows:

The TIP has a pair of output buffers of equal size, say 800 bits (100 characters) for each terminal. The TIP initially allocates to the sending Host 1 message and 800 bits. One of the buffers is always being used to output to the terminal while the other is used to accept data from the sending Host.

Let's call the buffer currently pointing at the terminal the Tbuf and the buffer currently pointing at the network the Nbuf.

When the first message arrives from the network that data is put into Nbuf.

The buffers are toggled and a new allocation is sent of 1 message and "L" bits where "L" is the length of the previous message. The buffer just filled, now Tbuf, is output to the terminal. When the next message arrives from the network it is put into Nbuf.

When Tbuf is empty the buffers are toggled and a new allocate is sent of 1 message and "L" bits.

We do not understand why the message allocation is limited to one at a time. It seems to us that it would be possible to allocate several messages and append the data that arrives to the data already in Nbuf.

TIP allocation works as it does for a number of reasons. First, the double buffering scheme the TIP uses is somewhat simpler to implement than would be the obvious alternative of a circular buffering scheme. Further, the TIP's implementation assumes that the sending Host, if trying for high throughput, will fill each message to the allowable bit allocations. This should be no problem given the small size of the TIP buffers (this assumption is false to the extent that a Host's NCP implementation will not permit users to control the fullness of transmitted messages). For the TIP to use a circular buffering scheme would add some words of code, perhaps only about 100, but these words would affect performance since they would be in the inner loop (end tests are harder in a circular buffering system than in a double buffering system since in a circular buffering system the boundaries between messages vary in their position in the buffer and one must be concerned with wrapping around the end of the buffer). Also, using a circular buffering scheme instead of a double buffering system would require memory of exactly how many outstanding bits of allocation there are (something that is implicit and takes no memory in the present implementation so

that retransmission of ALLocates could be done correctly (this would require an approximately sixty-four word table to store the additional bit allocate memory). Given the small size of the TIP buffers already, it is not clear that the additional loss of memory to go to a circular buffering scheme would be compensated for by the advantages that a circular buffering scheme offers, primarily in the area of less fragmentation of the available buffer space. Given sufficient buffering in the TIP and a Host pushing as hard as allowed by the TIP, double buffering should be able to hide any network (and other) delays. A fundamental problem, given long distances across the network and through the Hosts, is not so much the TIP's allocation strategy as the lack of sufficient TIP memory for buffering. Any allocation strategy would work poorly with the small amount of buffering available on many TIPs.

Finally, assume that the available TIP buffers for a port are well matched to the speed of the terminal connected to the port and to the network round trip time (i.e., printing half the buffer takes the same time it takes for an ALLocate to go to the sending Host and for a data message to make its way back across the network to the TIP -- it is unlikely that the TIP will have more than this much buffering due to its very limited buffering capacity). In this case, the TIP's buffering and allocation

strategy is optimal. The sending Host cannot in general know the speed of the terminal to which it is sending. Therefore, the only clue the sending Host has available about when the TIP is ready for more data is receipt of an ALLocate. As was mentioned above, if the sending Host follows a general policy of sending messages of a length less than half the total available buffer space, smooth (i.e., maximum rate) output cannot be achieved, because the TIP quickly prints the data in the small message it has received and has to sit idle while the ALLocate travels from the TIP to the Host and a data message travels from the Host to the TIP. Possibly less obvious is the fact that if the sending Host is allowed to send messages with length greater than half the available TIP buffering space and the Host frequently follows a policy of sending such larger messages, smooth (i.e., maximum) throughput again cannot be achieved, because after the TIP has printed the large message it has nothing to print while the ALLocate is sent to the Host and more data is sent back to the TIP. The implication of these facts are that it is difficult to find a simple strategy of sending allocations which permits the sending Host to effectively take advantage of the incremental possibilities of a circular buffering strategy. It is true that if for one reason or another the sending Host cannot generally send the maximum data allowed by the TIP's allocation, then the

TIP's double buffering system is less flexible, from the point of view of the sending Host, than would be a circular buffering system.

For these reasons, we have decided not to change the TIP's buffering and allocation strategy.

2.1.6 TIP Bandwidth (i.e., Throughput Capacity)

We have recently spent a significant amount of effort attempting to understand the TIP's bandwidth (or throughput capacity). We have counted instructions in the TIP to discover the cost of various TIP functions and we have constructed mathematical expressions which relate these costs to the various parameters of the system (e.g., number of terminals, mix of terminal speeds, available buffering, efficiency of Host packing of data into messages, average number of ALLocates packed into a single control message from the TIP to a Host, etc.). Unfortunately, the results are very sensitive to the choice of values for the various parameters. For instance, the capacity of a TIP may be less than 10Kbs in a configuration with sixty-three ports all doing output at 150 baud with each message for these ports containing only one character. On the other hand, if the ports are running at 1200 baud and the size of the messages is increased by a factor of eight, the TIP's capacity may be four

times as great. Other mixes of parameter values result in still higher capacity results. Our present problem then is to turn our mathematical expressions into graphs which provide insight into what capacities the TIP is capable of and where on the curves any given site is running.

There have been some immediate results of our TIP capacity investigations. For example, we have confirmed (as we always suspected) that the TIP's code for sending ALLocates is very costly to run. Further, because of the TIP's allocation strategy, this code is run very often (approaching once for every message that is received); and because of the TIP's limited buffering capacity, messages are received very often. We are studying methods of reducing the cost of the TIP's ALLocate sending mechanism. This is an example of a vicious cycle in which the TIP sometimes finds itself. Because the TIP has limited space, its data structures and code tend to be optimized to take little space and therefore generally are very costly to use. But because of the TIP's limited memory for buffering, the very costly (in terms of throughput capacity used) code must be run very often.

Also as a result of our capacity investigations we have discovered that the TIP has been looping through the IMP

background loop more often than absolutely necessary, resulting in a 15% decrease in TIP capacity in some cases. Further, we have discovered that the TIP's capacity is effectively reduced by certain IMP computations which take an inordinate fraction of the computer. Steps are being taken to correct these problems, and we will continue to be on the lookout for other such problems.

2.1.7 Conclusion

We have found a number of problems and sub-optimalties in the performance of specific TIPs and of the TIP system in general. However, we have found no fundamental problem with the TIP's design or the way it has been operating, given the constraints of memory limitation, average network path length, and typical Host delay.

2.2 IMP Performance and Improvements

Our previous Quarterly Technical Report discusses changes made to the IMP program which reduce interference between Hosts on an IMP, restructure the source-to-destination IMP message number mechanism to expand it and make it more reliable, and do more accurate packet buffer accounting. All of these changes act to improve the IMP's performance. In the remainder of this subsection, we discuss a number of other specific steps being taken to improve the IMP's performance.

2.2.1 Memory Space

We first outline the present situation, as shown in Figure 1. The IMP has available to it 16 kilowords of memory with packet buffers scattered throughout memory; there are at most 43 packet buffers. The Very Distant Host (VDH) code occupies an additional 1.2 kilowords of memory, reducing the space available for packet buffering by 35 percent.

In either case, there are sufficiently few packet buffers as to cause visible throughput limits and frequent delays. For instance, the throughput limits can be seen from the following sort of reasoning. Assume that only ten packet buffers are available for some function such as message reassembly. Further assume that a buffer is in use for each packet for an average of 100 msec. Then, if each buffer can hold about 1000 bits, the maximum reassembly rate (or throughput to a destination Host) would be

$$\frac{10 \text{ buffers} * 1000 \text{ bits/buffer}}{100 \text{ msec}} = 100,000 \text{ bits/sec.}$$

Delays happen, for instance, in the following way. In certain circumstances, the source IMP must request of a destination IMP an allocation of enough packet buffers to reassemble an entire message. With fewer buffers total, the

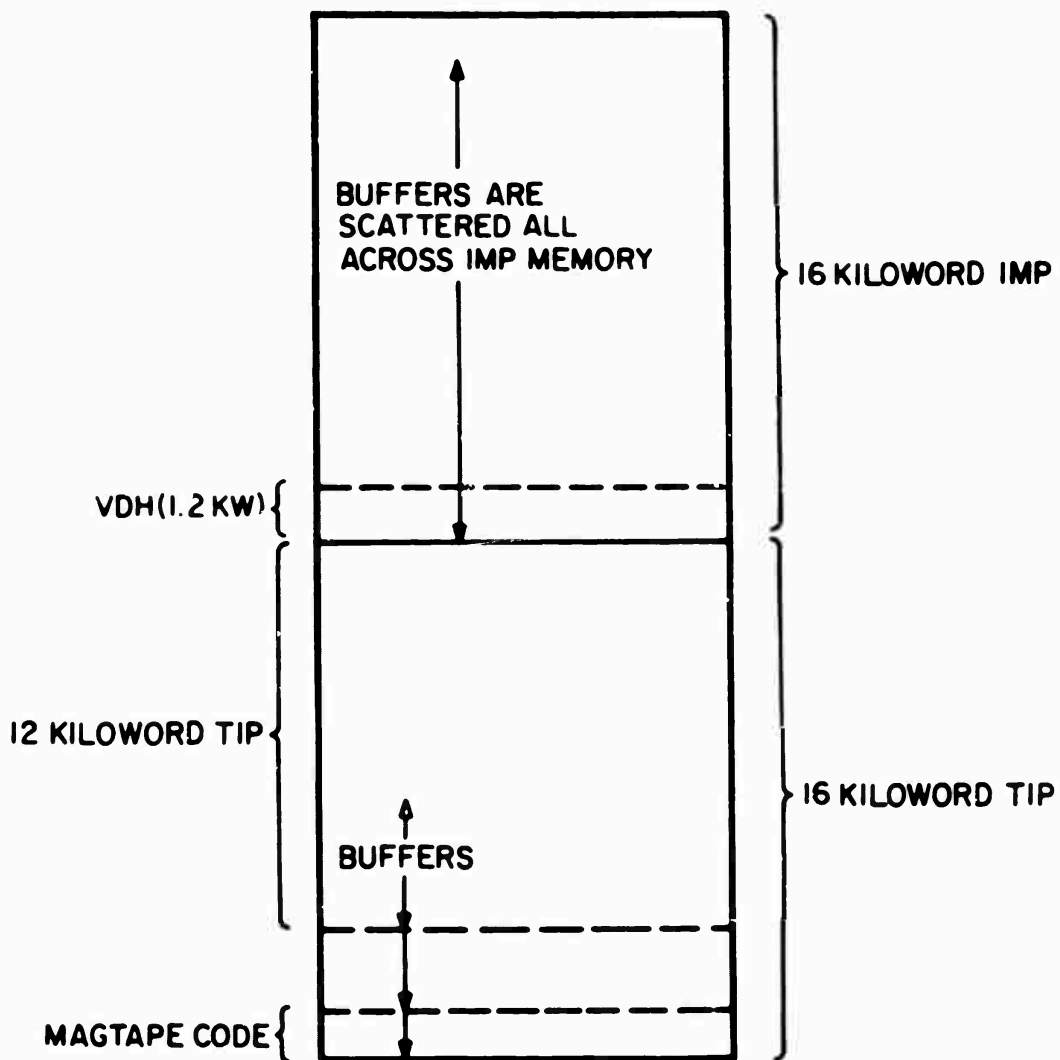


Figure 1 -- IMP/TIP Memory Layout

destination IMP will with higher probability not immediately have available the requested buffers and some delay will be incurred waiting for the necessary buffers to become free.

We have considered a variety of potential methods for relieving the buffer space problem in the IMP. These methods fall into four categories: a) change the IMP program so it can use memory (as it cannot now do) above the 16 kiloword boundary; b) utilize existing buffer space more efficiently; c) remove some of the present code or data structures; and d) move the VDH code out of the 16 kilowords of IMP memory. Option a is quite difficult, requiring probably two man months to accomplish, slightly more code than before, and an opportunity to use the newly accessible memory only on TIPless IMPs where somebody has purchased additional memory.

As for option b, utilizing existing buffer space more efficiently, two ideas have been considered. The first is to go to a bi-modal buffer size which would effectively double or quadruple the number of store and forward buffers at the expense of some processing efficiency and some additional code. The second idea is to provide for allocations of less than eight packets in the case of multi-packet messages, thus effectively increasing the buffers available for reassembly in some cases.

This latter idea would be at the expense of considerable complexity and some extra code.

Option c has many possible subparts some of which we enumerate below:

- i. The IMP's initialization code takes about 800 words of memory and is probably run on the average of 10 more than once per week. Retaining the initialization code allows IMPs to be restarted. Removing the initialization code would result in eight more packet buffers but would require a reload from the NCC of a completely initialized core image every time the machine was to be restarted. The tradeoff is clear: operational convenience against more space for packet buffers.
- ii. We have considered removing the relatively complete IMP DDT program in favor of a trivial inspect and change capability which could be operated remotely by a fancy DDT which resided at the NCC.
- iii. We have considered removing the IMP's capability to handle more than four inter-IMP circuits.

- iv. We have considered removing the IMP's statistics package.
- v. We have considered removing one of the IMP's loaders (it has two) which would make it slightly more difficult to reload a dead IMP in some cases.
- vi. The IMP's packet reloading mechanism is somewhat more bulky than need be. We plan to revise it to make it smaller.
- vii. We have considered removing the IMP's capability to handle a local Teletype.
- viii. The IMP's reassembly block structure is somewhat more bulky than need be. We plan to revise it to make it smaller.
- ix. In the Pluribus IMP we have developed a system of "transaction blocks" which are used to keep track of outstanding messages. An ad hoc system which does not use transaction blocks is presently used in the 316 IMP, and this ad hoc system is somewhat bulky. We plan to convert to the transaction block system in the 316 IMP.

Solutions to the buffer space problems based on options a, b, and c are still in the formative stages. Option d, moving the VDH out of the IMP memory, is somewhat further along.

In the case of TIPs, option d means moving the VDH into the TIP space which will be harmful to the TIP's buffering capacity. While with a full 32 kiloword memory, TIPs will in many cases have adequate buffering, it is clear that combinations of terminal types and numbers of terminals can be attached to a TIP which the TIP will have insufficient memory to buffer adequately. In this case we see but two solutions: a) limit the numbers and rates of terminals on a given TIP to values which can be buffered adequately; and b) reconfigure the TIP so it has its own machine, with all the available memory, independent of the IMP. In the case of an IMP without a TIP, it is a simple and relatively inexpensive matter to add additional memory to the IMP in which the VDH code can reside. On ARPA's instructions, we have begun to make the necessary modifications to the VDH code to allow it to run in the additional memory, and this task will be finished early in the third quarter.

2.2.2 One-packet Turbulence

A problem which caused single packet message stream throughput degradation (described in QTR 7, Section 7.5) has been

fixed. Originally, if a single-packet message arrived out of order, the destination IMP would return an ALLOCATE instead of a RFNM message. If single-packet messages were being sent at a sufficiently rapid rate by the source Host, succeeding messages would be sent off before the arrival of the previous ALLOCATE, and a quasi-stable state would persist where every message had to be retransmitted. A return to more normal operation would occur only when the source Host would slow down enough for the ALLOCATE of the last outstanding message to return before a new message came in from the Host. A temporary solution was implemented which detected the undesirable state and then stopped accepting messages from the source Host until all outstanding messages were completed. This solution, however, led to a more "bursty" type of throughput degradation when a message stream made use of load splitting, i.e., concurrent alternate routes through the network. Under such circumstances, out-of-order packets are quite frequent and each time they occurred the message stream was held up for at least the round-trip time of a message from the source IMP to the destination IMP and back. The current solution goes back to the original cause of the problem: rejecting out-of-order single packet messages. The main reason for doing so was to avoid a reassembly lockup where in-order messages could not be accepted or allocated because the space was already occupied by out of

order messages. With this consideration in mind, the current solution is to accept out of order messages, but to keep them on a separate message stack; thus, should a potential lockup situation arise, any one of the out-of-order messages can be reclaimed for storage; an ALLOCATE will eventually be sent and the message retransmitted, in proper order this time. With this solution, throughput degradations due to out-of-order messages occur only when there is a drastic lack of storage at the destination IMP, and the degradation only lasts as long as such a lack of storage persists.

2.2.3 Eight-packet Message Congestion

A problem which occasionally caused multi-packet message stream throughput degradation has been fixed. The problem occurred when a Host which was not RFNM-driven (i.e., did not wait for a RFNM before sending the next message) exceeded the buffering capacity at the destination IMP, due to either, or a combination of, low reassembly space at the destination IMP or high network delay. Because the destination IMP would wait for some time before sending back the RFNM in the hope of being able to piggyback a new ALLOCATE for eight packets, additional REQUESTs for eight-packet allocates would be generated by the source IMP. A situation would develop where the destination IMP

would have a queue of replies consisting of intermixed RFNMs and ALLOCATEs, using most or all available outstanding message number slots. The effects of this situation are twofold. One is that Host throughput to the congested destination is degraded due to lack of available message number slots, and in fact Host throughput to all destinations is stopped if the Host interface is blocked waiting for a message number slot. The other effect is that RFNMs are unnecessarily delayed in being sent back to the source IMP, with possible adverse effects on the source Host. The solution is to modify the process at the destination IMP which sends off the replies to messages (RFNMs, ALLOCATEs, etc.). If a RFNM for an 8-packet message is to be sent, an attempt is made to piggyback an ALLOCATE of eight on the RFNM message. If the storage cannot be immediately allocated, sending the RFNM is delayed in the hope that such storage will become available in short order. The RFNM is sent off without the piggybacked ALLOCATE either after a 1/2 second of waiting, or if another RFNM or ALLOCATE for an 8-packet message is waiting to be sent for some later message number. This next RFNM or ALLOCATE is just as capable of carrying back the desired ALLOCATE. Using this new scheme for deciding when to send back RFNMs cuts down on the delay for RFNMs when other messages are outstanding. Since these RFNMs tend to come back without ALLOCATEs, the use of message

number slots by the source IMP is regulated by the number of 8-packet reassembly areas in the destination IMP.

2.2.4 Routing Problems

Another modification to the IMP program was improving the routing algorithm's hold down of the delay path (see QTR No. 4, p. 4). The modification consisted of two parts. As the first part, we fixed the hold down mechanism so that when routing is to go into hold down when it was already in hold down, the hold down timer is reset to its maximum value. This assures that each time the criteria to go into hold down are met, the full hold down cycle occurs.

The second part of the hold down modification was to correct a problem in the implementation of the criteria for entering hold down. Previously, hold down was entered if the delay difference, between the delay estimate to a given Host in an arriving routing message and the delay estimate that the IMP was previously maintaining, was greater than a certain value (twice the minimum per-hop per-complete-nominal-routing-period delay increment). However, because of the possibility of changes in the delay estimate based on an interval of time less than the complete nominal routing period (because of the possibility of routing being sent more frequently than the rate of the nominal routing

period), it was possible for the delay to increase in a number of incremental steps over several sub-routing-periods to a total which should have caused hold down to be entered, but hold down was not actually entered since no one increment was sufficient. The correction was to maintain a sliding window one nominal routing period wide over which incremental delay increases are summed such that if the sum exceeds the value mentioned above, hold down is entered. Thus, it is now difficult for routing delay changes to sneak, in small steps, past the criteria for entering hold down.

The problem with hold down was first noticed by the Network Measurement Center which observed packets destined for a particular IMP looping for a duration equal to the hold down period between two neighboring IMPs which were holding down the path to the destination. Once the changes were implemented, packets no longer looped significantly.

Also in the area of routing, we have observed empirically in recent months a problem which appeared to be related to variable frequency routing transmission. The network routing algorithm has for some time now had the capability to transmit routing information at two, three, four, or five times the nominal routing frequency. The motivation for this capability is to

permit more rapid propagation of routing in cases when the network circuits are lightly loaded and there is no better use for the circuit capacity than to transmit routing. As the network has grown bigger in recent months, we have noted that when routing was only transmitted at the nominal frequency, problems with routing occurred, most often of the form of not all nodes noting when a given node went down and came back up quickly, the information about such downs being part of routing transmissions. The explanation for this trouble has been found to be that because of the present large size of the network (i.e., the maximum path lengths are quite long), routing at only the nominal frequency does not always result in routing being propagated fast enough to cover the maximum necessary distances (this is a probabilistic rather than deterministic effect). However, with the routing frequency greater than the nominal value (which is the normal case since the network circuits are normally relatively lightly loaded), the above mentioned problem is not seen; i.e., routing propagates fast enough to cover the maximum necessary distances when a higher than nominal routing frequency is used.

2.2.5 IMP Bandwidth

As part of our IMP performance study, we redid our analysis of the processing required to handle messages in the network. This was done by actually counting the machine cycles on the various program paths. While we calculated the bandwidth of both the 316 and 516 versions of the IMP (the 516 version has a slower I/O data channel, but has faster instructions), in this section we consider only the 316 version, letting the 516 version rest with the statement that that version generally has somewhat more bandwidth than the 316 version. We justify this omission of the details of 516 bandwidth on the grounds of simplifying our presentation and in light of the relatively few 516 IMPs in the network.

Figure 2 shows the present maximum message processing bandwidth of a 316 IMP for four different types of traffic. In addition, we can use this maximum bandwidth information along with observed IMP behavior and recorded network traffic characteristics to estimate the current average IMP message processing capacity and the average load imposed on this capacity (for the purposes of these calculations we assume the network configuration of May 1975).

We have observed that there is an approximately 18% degradation in Host processing bandwidth in the IMP for each

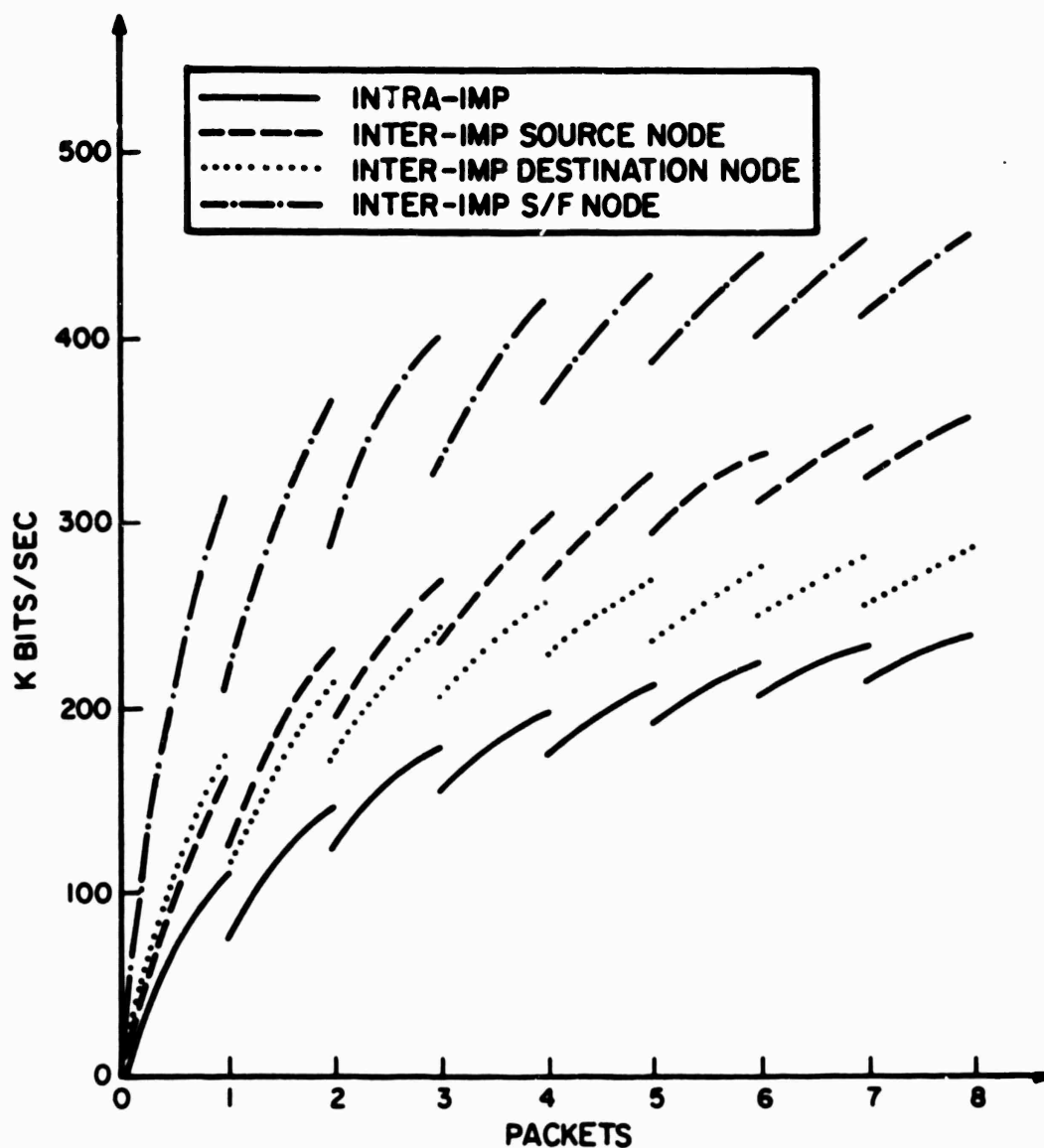


Figure 2 -- 316 IMP Message Processing Bandwidth

active inter-IMP circuit connected to the IMP because of the burden and interference of routing processing (assuming routing at the maximum frequency of five times the nominal frequency). It is fair to say that this same degradation would apply to store-and-forward processing since inter-IMP I/O is about equivalent to Host I/O from the point of view of routing processing. Another approximately 11% degradation has been observed when a TIP is running along with an IMP.

Taking into account that the average IMP has 2.24 inter-IMP circuits, one can deduce from the curves in the figure the actual average maximum IMP bandwidth at present. We now see why it would be particularly desirable to route with lesser frequency; however, as mentioned in the previous section, this has problems of its own.

Again taking into account the actual present degradation because of routing, and taking into account a) that 32.5% of the network traffic is intra-IMP, b) that the average network path length is 6.22 hops (and that network traffic therefore undergoes considerably more store-and-forward processing than other types of processing), c) that the average message length is closer to 240 bits than to the maximum, and d) that the average 24-hour network traffic is 1.2 Kbps for intra-IMP messages and 2.5 Kbps

for inter-IMP messages, and using a factor of 5 for peak-to-average loading, we can deduce from the curves that some nodes may be close to saturation for intra-IMP traffic but have reasonable spare capacity for inter-IMP traffic. Again, it would be very desirable to do routing less frequently.

In addition to pure bandwidth constraints, IMP throughput performance is sometimes constrained by timing considerations; for instance, if during the time a packet is being input from a Host, no processing can take place (i.e., the machine effectively sits idle waiting for the I/O to finish), and if the processing and I/O both take elapsed time (although the I/O takes very few actual machine cycles), then receiving and processing a packet from a Host takes twice as long (and effectively halves the throughput rate) as would be taken if I/O and processing could be overlapped. In fact, there is such a constraint on Host input processing in the IMP and this constraint affects the maximum throughput rates that can be obtained intra-IMP. The way to remove this constraint is to do double buffering on Host input (thus allowing processing to be overlapped with I/O), something that is not currently done.

2.3 New Developments

In addition to our efforts this past quarter to improve network performance, we have also begun planning and in some cases already begun implementing several new network developments. In the following subsections, we discuss each of these new developments in turn.

2.3.1 New TELNET Implementation

Implementation of the new TELNET protocol has now become the top priority item on the TIP development queue. Because what is done in the TIP potentially affects what is done at a number of Hosts, during the quarter we publicized the implementation schedule to which we hope to adhere. We summarize the schedule and its major milestones below.

Reasonably early in the third quarter we plan to be done with the basic design of the TIP's implementation of the new TELNET protocol, and to have a version of the TIP operational which has a general implementation (as opposed to the TIP's current very rudimentary implementation) of the TELNET option negotiator -- which, however, will refuse all options.

Late in the third quarter, we plan to have the design of the basic TELNET options done and to have operational a version of

the TIP which includes any modifications necessary to support these basic options.

Mid-way through the fourth quarter we plan to have a version of the TIP operational with the basic options operational; these basic options will be Binary Transmission, Echo, Suppress Go-Ahead, Timing Mark, and Remote Controlled Transmission and Echoing.

By the end of the year, we plan to have any problems worked out having to do with communication between the TIP's new TELNET protocol and other network Hosts and to have operational a solid version of the TIP which supports the new TELNET protocol with the basic options.

2.3.2 Expanding the Network Beyond Sixty-three Nodes

The network is now almost at its size limit of sixty-three nodes. To expand the network to greater than the current limit of sixty-three IMPs requires effort in several areas: a) specification of changes to the IMP/Host protocol (i.e., to BBN Report 1822) to permit addressing of greater than sixty-three IMPs, b) modifications to the IMP program, c) modifications to the Host Network Control Programs (NCPs), and d) modifications to the Network Control Center (NCC). Interestingly, no change is

necessary to the Host/Host protocol as the Host/Host protocol separates the IMP/Host protocol (and hence the addressing fields) from the rest of the Host/Host protocol. We discuss each of the above four areas in greater detail in the following paragraphs.

2.3.2.1 IMP/Host Protocol Changes

The IMP/Host protocol currently permits addressing of a maximum of sixty-three IMPs with four genuine and four "fake" Hosts on each IMP. The fake Hosts are the IMP debugging, statistics, and other software packages. Thus, the current address fields contain a total of nine bits of address information, six bits for IMP number, one bit to specify whether the Host is fake or not, and two bits to specify one of the four genuine or fake Hosts on an IMP.

We recommend expansion of the IMP/Host protocol address fields to 24 bits, 16 bits to address about 65,000 IMPs and eight bits to address about 250 genuine Hosts and a few fake Hosts on an IMP.

At the time one is making such a fundamental change to the IMP/Host protocol as expanding the address fields from nine to 24 bits, it is probably sensible to make a number of other protocol changes that have been requested for various purposes, and we are

currently evaluating these requests. We plan to make these changes in a backward compatible manner, so that Hosts which use the present IMP/Host protocol can continue to use it over an indefinitely long transitional period (of course, without any capability to communicate with Hosts whose addresses are outside the current limits). The details of the IMP/Host protocol change are discussed in Section 2.3.3.

2.3.2.2 Modifications to the IMP

The fundamental change necessary to the IMP software is to expand the network packet format to accommodate the expanded IMP and Host address fields. This is a significant but not too difficult change which can be done in a manner completely transparent to the Hosts. That is, the packet format can be expanded immediately with the change to the IMP/Host interface not made until some later time.

Of course, eventually the IMP/Host interface must be changed to follow the new expanded IMP/Host protocol. This should be done in such a manner that the IMP maintains both the old and the new interface, so that the IMP is able to communicate with Hosts following either the old or new protocol. Again this is a significant but not too difficult change.

Before the sixty-fourth IMP can actually be added to the network, some change must be made to the routing algorithm. This can be done in two steps. As an interim step one can just expand the current algorithm to permit a few more nodes (e.g., 75). Ultimately, the routing algorithm must be changed to permit area routing. Without area routing, as the number of nodes approaches twice the present number, the CPU bandwidth and the line bandwidth used by the present routing algorithm will become excessive. With an area routing algorithm, the CPU and line bandwidth required can probably be made to increase as the log of the number of nodes rather than linearly with the number of nodes.

The biggest problem with the IMP program changes is likely to be the memory required to implement them. The additional memory required will probably not be very much, but with the current insufficiency of IMP buffer memory, any additional memory taken for program changes will be painful.

2.3.2.3 Modification to the Host NCPs

From our view point, the necessary modification to the Host NCPs comes in two parts: the modifications necessary to the TIP and the modifications necessary to other Hosts. We discuss the TIP first.

There is a basic choice to be made for the TIP: should it or should it not be modified to permit communication with IMPs with addresses greater than sixty-three and Hosts with addresses greater than four (counting from one)? As stated above, the IMP should be fixed so that a given Host may either continue to use the old protocol (with its limited address fields) or use the new protocol (with its expanded address fields). Let us assume that the TIP will be modified to follow the expanded protocol.

First, the TIP's internal tables of Host addresses (i.e., the Host number with which a given terminal is communicating) must be expanded to 24 bits. This is straightforward, costing the memory for the additional bits for each of the possible TIP ports. Additionally, the program will have to be changed or added to at a number of points to utilize these expanded tables. Again this is straightforward but costs some memory. So far, the tables we are discussing are all tables which are indexed by TIP port number and are thus only sixty-four elements long. A more difficult change is required for the several tables which are presently indexed by Host number and are thus 256 elements long. Clearly, the basic structure of these tables must change as one cannot consider tables $65,000 \times 256$ (the number of IMPs times the number of Hosts per IMP) elements long. Thus, it will be necessary to make these tables more dynamic than they are at

present with entries only for Hosts with which TIP ports are communicating (a maximum of sixty-four) or to somehow change the Host protocol implementation to eliminate these tables which are presently indexed by Host number. We will use both of the ideas suggested in the previous sentence. Some of the TIP tables currently indexed by Host will be made more dynamic, such as the tables which keep track of Host RSTs, ERPs, and ECHs. The remainder of the TIP tables currently indexed by Host, primarily the table which keep track of blocking on the control link to the Hosts, will be eliminated in favor of a more clever implementation in which they are not required. These changes are likely to result in more CPU bandwidth being consumed than was consumed previously, although less memory might be required.

The NCP changes to Hosts other than the TIP we are less able to estimate. Our guess is that there are a number of NCPs which will have nearly the same structure as the TIP (i.e., some tables which need simple expansion and some tables which need to be made more dynamic). Of course, these other Hosts will also have the option of not changing at all and accepting the limitation of being able to address only sixty-three IMPs. Note that this option may be quite viable as most of the ARPA research sites will probably continue to have addresses of less than sixty-four (unless ARPA is planning to add a lot of new contractors to the

network), and will thus be able to communicate with each other, and probably have little interest in communicating with other Hosts (such as the Navy sites). Further, new sites which will have addresses greater than sixty-three can have their NCPs implemented so they can address the expanded set of IMPs and Hosts and will therefore be able to communicate with every site. (A little shuffling of IMP numbers might even better separate the ARPA and non-ARPA sites into less-than and greater-than groups.) Also, as these new sites are added, in some cases NCPs for existing Host types will be used and expanded, and these expanded NCPs can then be retrofitted to the Hosts at sites less than sixty-four. Finally, modification to a very few NCPs (e.g., TENEX, TIP, ELF) will result in over 50 of the existing Hosts being able to use the expanded network.

2.3.2.4 NCC Modifications

The major NCC efforts required to permit greater than sixty-three IMPs are in the areas of modification to a number of the NCC operator procedures and support programs which exist on the PDP-1 and BBN-TENEX and modifications to the program which runs in the 316 NCC computer. These changes are straightforward and can probably be done in a natural and evolutionary manner.

There may be facilities (e.g., the Network Measurements Center) which would have to undertake changes similar to those necessary for the NCC.

2.3.3 Proposed IMP/Host and Host/IMP Protocol Change

In the previous section, we discussed the need to expand the network beyond sixty-three nodes including, in general terms, the need to expand the IMP/Host and Host/IMP protocols. In this section we discuss in detail the change we propose to make to the IMP/Host and Host/IMP protocols, both for network expansion and other reasons. The information contained in this section was circulated throughout the ARPA Network community at about the mid-point of the second quarter. There is necessarily considerable overlap between the information given in this section and the issues raised in the preceeding section.

Our intention in this expansion is to correct certain existing limits without fundamental changes in the philosophy of the IMP/Host protocol; i.e., while many issues which would represent fundamental changes to the IMP/Host protocol are presently under discussion in the world-wide packet-switching community, we are not able to undertake massive fundamental changes on a time scale compatible with the short term needs for network improvement (e.g., there are already almost 60 IMPs).

The following paragraphs cover each of the major characteristics of the expanded protocol. A knowledge of Section 3 of BBN Report 1822 is assumed. As is discussed below, the expanded protocol is backwards compatible.

2.3.3.1 Expanded Leader Size

The leader will be expanded from two to five 16-bit words. This will provide space for necessary field expansions and additions.

2.3.3.2 Expanded Address Field

The address field will be expanded to 24 bits, 16 bits of IMP address and 8 bits of Host address. This expansion is more than adequate for any foreseeable ARPA Network growth.

2.3.3.3 New Message Length Field

A new field will be added which will allow the source Host to specify, if it wishes, the message length (in bits) to the IMP subnetwork. The IMP subnetwork may be able to use this information (when available) to better utilize network buffer storage. The destination Host may also be able to use this information to better utilize its buffer storage.

2.3.3.4 Expanded Handling Type Field

The handling type field which is now used to distinguish between priority and non-priority message streams, etc., will be expanded to eight bits. This expanded field will provide for the possibility of a number of parallel message streams having different handling characteristics between pairs of Hosts; e.g., priority, non-priority, varying numbers of packets per message (see below), unordered messages (i.e., the present type-3 messages), a message stream requiring guaranteed capacity, etc. Note that only some of these facilities will be available in the near term.

2.3.3.5 Source Host Control of Packets per Message

The possibility will exist for the source Host to specify a message stream which will use a given number of packets per multi-packet message (e.g., two packets per message or five packets per message). Since the IMP network will not have to use eight packet-buffers for reassembly purposes, as at present, this may result in better performance for such services. This will help users who need both low delay and high throughput.

2.3.3.6 Unordered (type-3) Message Change

Unordered messages may be indicated by a handling type rather than by a message type as at present. This would be compatible with the need to check the Host access control capabilities of all messages. It would cause a slight backward incompatibility for the three or so Hosts which presently use type-3 messages in their research.

2.3.3.7 Change in Format of Fake Host Addresses

The For/From IMP bit will be eliminated. The fake Host addresses will be the four highest Host numbers (e.g., IMP Teletype will be Host 252).

2.3.3.8 Addition of a Parameter to the IMP-to-Host NOP

The IMP-to-Host NOP will have added to it a parameter specifying the address (IMP and Host number) of the Host.

2.3.3.9 Backward Compatibility

The old and new formats will be supported in parallel in the IMPs for the foreseeable future to allow gradual phaseover of Host software. A Host will be able to specify to its IMP whether the old or new formats are to be used; thus, it will be possible for the Host to specify switching back and forth between the two

modes for debugging purposes. The specification of the mode to be used will be possible via a proper choice of format in the Host-to-IMP NOP message; the IMP will use the mode of the Host-to-IMP NOP message the IMP has received. Further, a Host may use either the old or new format without needing to know more about the other format messages than to discard them should they arrive. The IMP will initialize by sending several NOP messages of each type to give the Host its choice. Although a Host not implementing the new format will not be able to address Hosts on IMPs with IMP-number greater than sixty-three, the IMPs will wherever possible do the conversion necessary to permit Hosts using the old format to communicate with Hosts using the new format and the reverse. Finally, it will be possible to convert the leader format from old to new, or the reverse, without knowledge of the message type.

2.3.3.10 Non-blocking Host Interface

A mechanism will be provided which allows the IMP to refuse a message from a Host without blocking the Host interface. This mechanism will permit the IMP to gather the necessary resources to send the refused message and then ask the Host to resend the message. Finally, the Host will be permitted to ask to be able to send a message, and be notified when it can, without the message having actually to be sent and refused.

2.3.3.11 Maximum Message Length

The maximum number of bits of data in a message may be reduced by a few bits.

2.3.3.12 Implementation Plan

We have received a good deal of feedback about the proposal given immediately above for revising the IMP/Host and Host/IMP protocol. The response has been mostly favorable with several areas of general exception. Areas of concern have included the length of the new leader fields (being inconvenient for 36-bit word Hosts), the possibility of making the new message formats better match proposed international standard message formats, and possible effects on the Host/Host protocol.

We are presently trying to integrate into our proposal some of the improvements which have been suggested to us. Once this is done, we will publish a revised proposal and an implementation schedule for making the necessary changes to the IMP software. We will distribute the implementation schedule and other necessary information (e.g., format details) in plenty of time so that Hosts desiring to use the new protocol as soon as it is available will be able to do so.

2.3.5 Netnews

Late in the quarter, stimulated by the need to announce several upcoming changes in network administration and network developments, ARPA asked that the TIP's NETNEWS capability (implemented via the RSEXEC) be improved to permit selective reading of news items rather than permitting display of news items only in reverse chronological order as has been the case till now. Further, ARPA asked that the TIP be modified to provide a herald, calling attention to the existence of new news items as they appear. By the end of the quarter, the news herald was operational although change of the content of the herald depends on almost manual means. Early in the third quarter the capability to view the news selectively will be operational, utilizing for the time being a modification of the MSG program developed at the University of Southern California's Information Sciences Institute, running under TIPSER-RSEXEC.

3. TIP ACCESS CONTROL AND ACCOUNTING

In our Quarterly Technical Report No. 6 (Contract No. F08606-75-C-0027) we reported on our design of a mechanism which ARPA requested to provide access control and user accounting for Terminal IMPs. In Quarterly Technical Report No. 1 of the current series we discussed several problems which resulted from installation of this mechanism in the network, and our proposed solutions to these problems; it was noted that the problems were primarily administrative rather than technical. Further, we noted that the access control and accounting mechanism had been disabled pending ARPA review of the need for the mechanism and the proposed solutions to the various problems. This review was carried out during the second quarter and resulted in a decision by ARPA to abandon all requirements for TIP access control or accounting.

In spite of this decision, we continue to believe that the design of the access control and accounting mechanism was sound and that, given a requirement for these features (of sufficient importance to make it necessary to solve the administrative problems), the mechanism would have performed well. In fact, we obtained several machine-months of experience with the mechanism

and encountered no technical difficulties. Accordingly we* have documented, in this section, the general and specific features of the design, with emphasis on the fact that the mechanism is a non-trivial example of computer resource sharing.

Our design starts with the basic fact that users (and administrators) of a small computer, in this case the TIP, will always desire more service than it can provide, but that in a network environment services can be provided to a small computer by one or more larger computers. In particular, because of its memory and bandwidth limitations, the TIP is incapable of providing its users with a sophisticated command language. The TIP has no space to hold tables of passwords or statistics on its usage; thus, the TIP has no capability for access control or accounting. The TIP cannot distribute operational information to its users, such as announcements of system changes. Further examples are readily available. What the TIP does provide is a relatively transparent, simple, flexible, and high performance interface between a terminal and the network. However, if access control, accounting, and other operational capabilities were to

*Our colleagues who implemented the TENEX RSEXEC portions of the mechanism have contributed substantially to this section of this report, portions of which previously appeared in their Final Report on Natural Communication with Computers (Volume III, BBN Report No. 2976), and are included here in the interest of a complete and coherent presentation.

be provided, it was necessary to devise a mechanism to obtain these capabilities elsewhere.

In the following sections we sketch a system of computer resource sharing which is able to effectively provide the TIPs in the network with a set of advanced capabilities. We also discuss the fundamental structures upon which our computer resource sharing solution rests, and we describe some of the capabilities which the system currently provides. Finally, we consider some deficiencies and ramifications of our solution.

We have used the term "non-trivial" to describe our system of computer resource sharing. Our system is non-trivial in the following senses: 1) several man-years of effort were expended in an actual implementation; 2) altogether some twenty-five computers are involved; and 3) the system is capable of being used operationally "around the clock." Further, the system of resource sharing which we have developed is broader than just the provision of TIP functions; the same concepts can be generally used to permit a computer or collection of computers to enhance the capabilities of another computer or collection of computers. Thus, this mechanism only begins to illustrate the potential of such resource sharing computer systems.

3.1 Development of the RSEXEC TIPSER System

Because the TIP functions in a large computer network, it was natural to consider the possibility of using another Host on the network to provide some of the capabilities missing from the TIP. Our first experiment in this direction was to provide a TIP "news" capability through which TIP users could be notified of events which affected their use of the TIP (such as a change in the way a TIP command worked or the release of a new TIP system). The TIP was given a new command named NEWS. When a TIP user executed the NEWS command, a logical connection was made from the TIP user's terminal to a process in a particular PDP-10 Host on the network. This process was programmed to send the latest TIP news over the connection to the TIP upon receiving a connection from a TIP terminal. At the end of the news, the process would break the connection to the TIP. Alternately, the TIP user could explicitly break the connection at any time. Either case freed the terminal for communication with other Hosts for other purposes. While no special effort was made to hide the fact that another Host was being called on to provide the TIP news function, the user did not normally have to be concerned with the fact that another Host was involved; the user had only to execute a TIP command and, in effect, the TIP printed the news. Thus, we had implemented a rudimentary example of resource sharing.

At the time of this initial experiment with resource sharing to enhance the TIP's capabilities, the Resource Sharing Executive (RSEEXEC) system also began to come into being. The RSEEXEC is an experimental, distributed, executive-like* system which acts to couple the operation of some ARPA Network Hosts. RSEEXEC is designed to provide an environment which allows users to access network resources without requiring attention to network details such as communication protocols and without even requiring users to be aware that they are dealing with a network. RSEEXEC is currently used both as an operational service facility and as a vehicle for exploring the technical problems of realizing an effective environment for resource sharing.

Development of RSEEXEC was motivated initially by the desire to pool the computing and storage resources of the individual TENEX Hosts on the ARPA Network. At the time, the TENEX virtual machine was becoming a popular network resource (at present there are fourteen TENEX systems in the network). Further, it was becoming clear that for many users, in particular those whose access to the network was via TIPs or other non-TENEX Hosts, it should not actually matter which Host provides the TENEX service

*In our terminology, an "executive" is that program or command language interpreter which a user uses to communicate with an operating system.

so long as the users could do their computing in the manner to which they had become accustomed. A number of advantages would result from such resource sharing. The user would see TENEX as a much more accessible and reliable resource. Because he would no longer be dependent upon a single Host for his computing, he would be able to access the TENEX virtual machine even when one or more of the TENEX Hosts were unavailable. Of course, for him to be able to do so in a useful way, the TENEX file system would have to span across Host boundaries. The individual TENEX Hosts would see advantages also. For example, some sites, because of local storage limitations, do not provide all of the TENEX subsystems* to their users. Because the subsystems available would, in effect, be the "union" of the subsystems available on all TENEX Hosts, previously limited Hosts would be able to provide access to all TENEX subsystems.

During the development of the RSEXEC system two observations were made: first, since many of the features planned for the RSEXEC were well matched to the desires of TIP users, it became clear that with some additional effort the RSEXEC system could provide TIP users with the sophisticated command language and

*In TENEX terminology, a subsystem is a program which runs in user mode but which is available to all users as if it were a basic part of the operating system.

other features they desired; second, because the RSEXEC was to be run on several PDP-10 TENEX systems, RSEXEC could potentially provide capabilities to the TIP very reliably. (With a single Host providing a function, such as the news service discussed above, there would be times at which that Host would be down when some TIP user required the function.) Thus, it would be possible through TIP use of the RSEXEC to obtain TIP capabilities superior to any the TIP could provide itself or that could be provided with the help of any single other Host. Our attempt at resource sharing was becoming less rudimentary.

3.2 Current TIP/RSEXEC Capabilities

A service program called TIPSER (for TIP SERVER), which currently runs (alongside other user programs) on three ARPA network TENEX Hosts, allows TIPs to make direct use of certain features of RSEXEC as a "virtual executive". Development of the TIPSER-RSEXEC system has been guided by the general philosophy that the TIP should be a transparent front end component supporting only terminal-device-specific functions and that access control, accounting, command language interpretation, and other "large Host operating system-like" functions should be handled by other more capable (larger) network machines.

The redundant implementation of the TIPSER-RSEXEC serves to distribute the load among the machines providing the service and to increase the accessibility of the service by guaranteeing that the service is available whenever at least one TIPSER-RSEXEC site is up. Some of the services provided to TIP users are listed in Figure 3. The relationship of users, TIPs, TIPSERs, and the RSEXEC is illustrated schematically in Figure 4.

Two mechanisms were developed to support the redundant implementation. The first is a "broadcast" initial connection protocol (ICP) to enable a TIP to connect to an available and responsive RSEXEC rather than to a particular one at a specific site. Using this mechanism, a TIP broadcasts requests for service to the known TIPSER-RSEXEC sites and then selects the site that responds first as the one to provide the service.

The second mechanism was developed to maintain multiple copies of the various information files (e.g., news and schedules) at the TIPSER-RSEXEC sites. This mechanism allows additions to these distributed information files to be initiated from any TIPSER-RSEXEC site and guarantees that the additions are incorporated into each file image in a consistent manner.

Having now briefly mentioned the capabilities currently available to the TIP through use of the TIPSER-RSEXEC, the rest

The QUIT command allows the user to leave RSEXEC.

The HELP, DESCRIBE, and SERVERS commands give the user information on the available functions, how each function works, and which sites run RSEXEC.

The LINK, BREAK, REFUSE, and RECEIVE commands allow the user to link to other users, break links from other users, refuse links from other users, and accept links from other users.

The FULLDUPLEX, HALFDUPLEX, and TIMECONSTANT commands allow the user to set various parameters of the system operation.

The NETNEWS command allows the system operations staff to announce information of interest to users; the GRIPE command lets users tell the system operations staff how they think the system is working.

The WHERE, WHO, and SITES commands let a user find the site at which a particular active user is running, list the active users at a set of sites, and find the sites at which a particular user is known.

The NETSTAT, HOSTAT, SCHEDULES, and TENXSTAT commands let a user ascertain such information as which hosts are up or down, the future down time schedules of IMPs and TIPS and various hosts, and the instantaneous loads on various of the network TENEX systems.

The TRMINF command allows the user to determine certain information about the TIP port he is using.

Figure 3 -- TIPSER-RSEXEC Command Functions

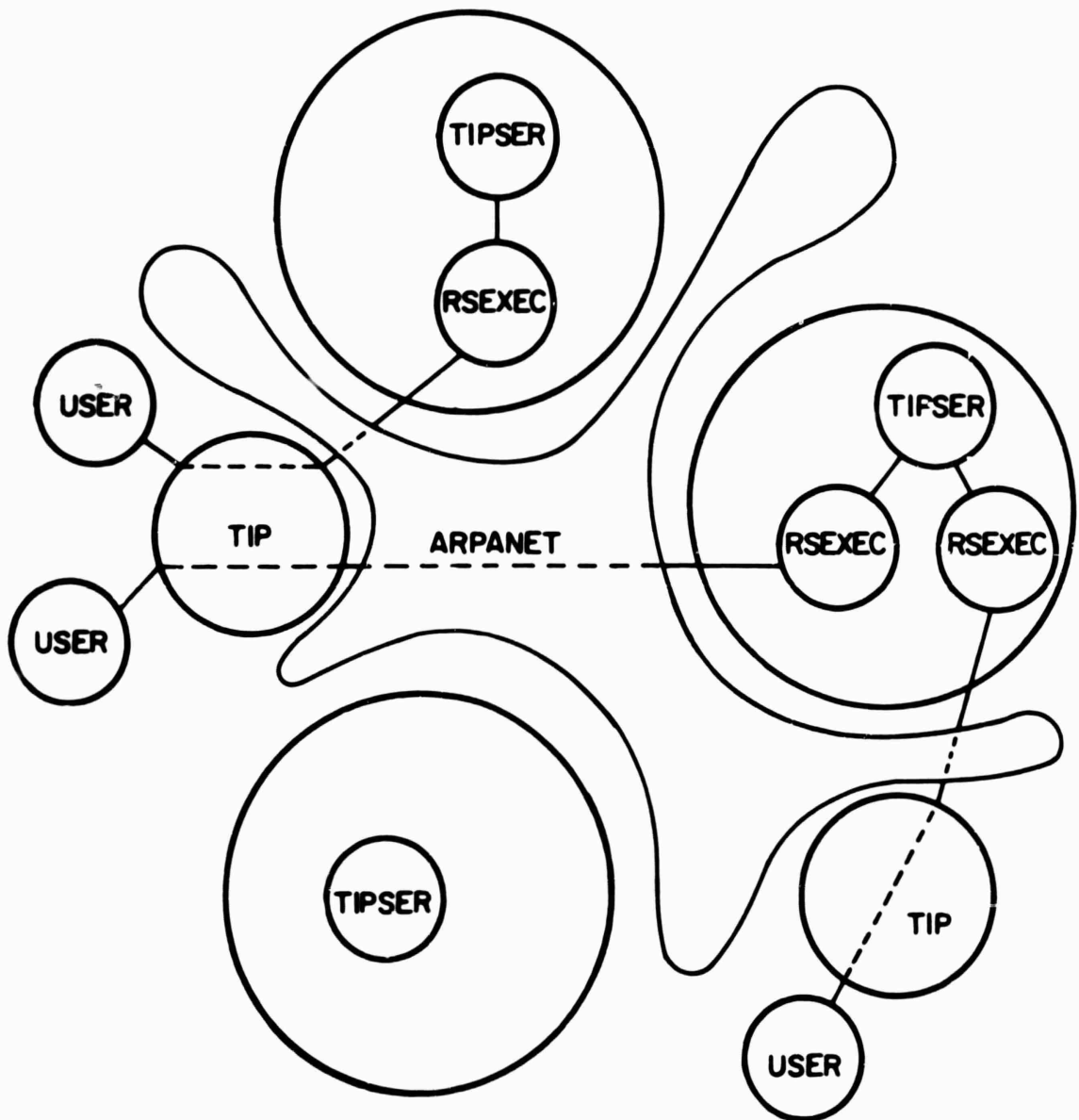


Figure 4 -- Relationship of Users, TIPS, TIPSERs, and RSEXEC

of this section describes in detail the pair of functions (TIP access control and accounting) which were provided using the techniques for resource sharing which we have developed.

In order to solve the problem of controlling access to the network and the related one of accounting for TIP usage, a distributed, multi-computer access control and accounting system for TIPs based on the TIPSER-RSEXEC and the RSEXEC distributed file system was developed. This system consists* of three distinct, but related, components: network login server processes (TIPSER-RSEXEC processes), data collection server processes, and data reduction software.

Whenever a user activates a TIP port, the TIP uses the broadcast ICP mechanism to connect to an RSEXEC which acts as a network login server. If the user successfully supplies a valid name and password, he is granted continued access to the TIP, the network, and to the standard TIPSER-RSEXEC functions. In addition, the RSEXEC transmits the user's network ID code (which serves to uniquely identify the user for accounting and subsequent authentication purposes) to the TIP and makes a "login" entry into an "incremental" TIP accounting data file. If

*We will describe this system in the present tense in order to avoid awkward English.

the user fails to supply a valid name and password within the allowed time, he is denied further access to the TIP.

After the TIP receives the user's network ID code it activates "connect time" and (outgoing) message counters to accumulate usage data for the user's session. These counters remain active until the user terminates his TIP session. Periodically the TIP executes an "accounting checkpoint" procedure whereby it transmits usage data, accumulated since the last checkpoint for its active users, to a data collection server process. The data collection server stores the checkpoint data in an incremental TIP accounting file for later processing.

Like the TIPSER-RSEXEC login servers, the data collection servers are redundantly implemented to insure high availability and to achieve load sharing. The TIP uses a request mechanism similar to the broadcast ICP to select one of the servers to accept its checkpoint data. The protocol used for this purpose is quite general and can be used for the collection of data other than that for TIP accounting. Furthermore, the protocol is designed to allow considerable flexibility in the choice of a server. For example, a TIP can switch from one data collection server to another after initially choosing one in the event that the chosen server can not complete the transaction (for example, because of network or Host failure).

The collection of incremental accounting files created by the data collection servers is a large, distributed and segmented data base. The reduction of data in that distributed data base to produce periodic accounting summaries is accomplished by software which executes within the environment provided by the RSEXEC distributed file system. This software performs a complex series of data management and network access operations in response to simple commands. When the "TIP accountant" (a human) issues the proper commands, the software automatically connects to the data collection sites and selectively retrieves and processes remote (and previously unprocessed) accounting data. This software was designed to be consistent with the RSEXEC philosophy: to allow a user to deal with resources (in this case accounting data) distributed throughout the network while relieving the TIP accountant of the complexities of dealing directly with the network itself.

We reiterate that the significance of the TIPSER-RSEXEC system exceeds the utility of the particular functions it currently supports. It has served to demonstrate the feasibility of having small Hosts share the resources of larger Hosts to reliably support features that exceed the small Hosts' own capacities. Users of a small Host obtain these services automatically in a network transparent manner.

3.3 Fundamental Structures

In addition to the standard communications protocols used by Hosts for communication between themselves, structures providing several additional functions were necessary to allow TIP/RSEXEC resource sharing. In the following subsections we discuss each of these structures, two of which have already been alluded to in the previous section.

3.3.1 Broadcast Service Requests

For a TIP user to be able to conveniently discover and use an available instance of the RSEXEC requires some mechanism other than the user simply trying to connect to each TIPSER-RSEXEC site in turn until an available one is found. This is a general problem of attempting resource sharing -- the problem of finding and selecting resources. Two techniques for supporting the selection function are apparent:

1. Maintain up-to-date status information about the various network resources and machines, and use it to select the machine best suited for a task. The server processes that support the RSEXEC system exchange status information for this purpose. Although automatic job assignment has not yet been implemented,

the status information is currently available to users who may use it to manually select a machine and is, in principle, available to programs for automatic resource selection purposes.

2. Dispatch "requests for service" to the appropriate machines, allowing them to respond with status information if they choose, and then make a selection on the basis of those machines which have responded as willing to accept a new task. This is the technique TIPs use when it is necessary to select a responsive RSEXEC.

The first technique involves a fixed overhead (i.e., exchanging and maintaining the resource status information) which is independent of the frequency of resource selection. For the second technique, the overhead is incurred on a per transaction basis and is, therefore, proportional to the frequency of selection. Although the frequency of service requests is relatively high in the TIP/RSEXEC case, the second technique is used because it does not require the TIP to allocate limited storage resources for maintaining status information. Another basic difference in these two techniques is that the second allows the constituent machines to retain a higher degree of

autonomy in managing their own resources. Each machine can choose whether or not to respond to particular requests for service.

3.3.2 Distributed Data Base Management

Multi-computer systems introduce a new class of data base management problems which result from the distributed nature of the data. These problems occur at all levels of system design and implementation, ranging from low level system primitives to function oriented application software.

Experience with the ARPA Network indicates that data tends to be distributed for a variety of reasons.

1. To insure reliability. The accessibility of critical data can be increased by redundantly maintaining it. The network user ID data base used by the TIPSER-RSEXEC to authenticate users is an example of a data base which is redundantly distributed to achieve highly reliable access.
2. To insure efficiency of access. Data can be more quickly and efficiently accessed if it is "near" the accessing process. A copy of the network user ID data base is maintained at each of the TIPSER-RSEXEC sites

to insure rapid, efficient access. (Reliability considerations dictate that this data base be redundantly maintained, and efficiency considerations dictate that a copy be maintained at each authentication site.)

3. As a consequence of the naturally distributed manner in which the data is generated or collected. The data base represented by the collection of incremental TIP accounting files is an example of a data base generated in this way. Individual data items are stored at the data collection site best prepared to handle them at the time they were generated by some TIP.

There are two fundamentally different types of distributed data bases. The first is one which is maintained "identically" at a number of sites. The second type consists of distributed, non-overlapping segments; that is, the data base is a collection of segments, each of which is singly maintained at a (possibly) different location. It is important to recognize that these two types represent extremes and that applications may call for "intermediate" types - for example, a data base consisting of a collection of segments some, but not all, of which are redundantly maintained.

The emphasis of our work within the TIPSER-RSEXEC context with the first type of data base has been to develop techniques for consistently and automatically maintaining the redundant data base copies. Below we cite two applications of such data bases and describe the techniques used in their implementation:

1. The TIPSER-RSEXEC maintains a copy of the TIP news file at each of the TIPSER-RSEXEC sites. Updates to the news file are limited to addition of news items. The system allows additions to the data base to be initiated at any TIPSER-RSEXEC site and ensures that all such updates are transmitted to and incorporated into all copies of the data base.
2. The TIP login system requires that the network user ID data base be maintained in a consistent manner at all TIPSER-RSEXEC sites. Each copy of this data base is a collection of mutually independent user entries. Allowable updates to this data base include the addition, modification, and removal of individual user entries. We have designed a data base management technique which allows updates to be initiated at any site and guarantees that they are consistently incorporated into all copies of the data base. By

"consistently incorporated" we mean that if all updating activity were to cease, all copies of the data base would eventually be identical.

The techniques used to maintain the NETNEWS and the user ID data bases each consist of two independent parts:

1. A reliable, data-independent, update transmission and distribution mechanism which uses persistent processes at the update entry sites to guarantee that all updates are eventually delivered to all data base sites (once, and only once).
2. A data-dependent update action procedure which is activated at data base sites whenever update commands arrive.

For the NETNEWS, the update procedure is a relatively simple one in which updates are appended to the data base as they arrive. For the user ID data base a more sophisticated update procedure is required. The nature of the data base and the operations permitted on it are such that recent updates to an entry override (rather than interact with) older updates. For example, when a user password is changed, the old password is simply replaced with the new one. The update procedure is based

on the use of a time stamping mechanism to enable each of the different data base sites to reconstruct and then act upon the (identical) sequence of update events. Furthermore, each entry (and modifiable subfield) in the data base retains the time stamp of the update which resulted in its current value. When most update commands arrive at a data base site, the command can be incorporated or rejected simply by comparing its time stamp with that of the data base entry to which it refers. The deletion and creation of entries require slightly special treatment. For example, if create and delete commands for a single entry are initiated at separate sites, network or system malfunction could cause the creation command to arrive at a third site after the deletion command. To properly handle such cases the data base update procedure defers "final" action on a deletion command until it is a certainty that all update commands for an entry which were initiated prior to the deletion have arrived. Only at that point is it safe to remove the entry from the data base.

The operation of the TIP accounting system results in the creation and manipulation of segmented data bases. The primary concern in the accounting application was with data base organization and convenient data access. The specific data base issues that required attention were:

1. Cataloging. It is obviously important to know where the various data segments (incremental accounting files) reside so that they can be accessed. The cataloging function is provided by the RSEEXEC distributed file system.
2. Insuring that no duplicate entries occur. Because the entries contain accounting information, it is critical that there is no redundancy. The data collection protocol was carefully designed to prevent the occurrence of duplicate data entries in spite of the broadcasting of data.
3. Insuring that each data base entry is processed exactly once when accounting summaries are produced. It is interesting to note that time stamping can also play a fundamental role in guaranteeing "once only" processing.

3.4 Discussion

Despite the fact that the system attained operational status, there were some clear deficiencies, and we have learned some important lessons. We also see some ramifications of the system on technical and operational aspects of the network and

network Hosts. Finally, we see almost unbounded potential for the use and growth of our system and systems like it. We discuss these issues in the rest of this section.

In a number of situations the existing ARPA Network Host-Host protocol forces difficult or clumsy implementations in support of functions which are conceptually quite simple. These difficulties are largely due to the complexity of the protocol. Typical situations which pose such difficulties can be characterized as involving brief, transaction oriented interactions. The TIPSER-RSEXEC broadcast connection mechanism is a good example of such a situation. The mechanism requires the transmission of a short message from a process to one or more remote processes. The standard Host-Host protocol requires that the processes participate in an elaborate exchange of protocol commands, carefully remembering the exact state of each exchange, in order for the first process to transmit its simple message to the other processes. For large Hosts this exchange is wasteful. For small Hosts it is often impossible to implement correctly. In this regard, it is interesting to note that the data collection protocol used in the TIP accounting system was designed to be separate from (and exist in parallel with) the Host-Host protocol in order to make implementation feasible for (memory) resource-limited TIPs.

The presence of multiple components in a distributed system, together with the potential for redundancy, makes it possible to achieve reliability by constructing systems from modules each of which is relatively simple. By using simple modules, component failure due to malfunction of non-essential features can be reduced. The evolution of the TIP and TIPSER-RSEXEC is a good example of this approach. Use of redundantly supported "logical" front end servers allows the network access machine to be simple and reliable without loss of function. The more complex "front-end-like" features can be reliably provided by network service machines rather than within the network access machine itself. Such a system takes full advantage of both the heterogeneity and homogeneity of various network components. The important issues in designing a system of this type are the assignment of functions among the various machines, the degree of redundancy required, and the protocols used to bind the system modules together.

Experience with the ARPA Network has indicated the potential need for access controls above and beyond those supported by the constituent Host service machines. For example, an access control mechanism has recently been implemented within the subnetwork to allow the set of network Hosts with which a particular Host can communicate to be administratively limited.

The access controls applied to the TIP also fall into this category. In many cases the goals of network transparency and ease of access conflict with those of security and privacy. Each security or access check places a barrier between the user (or his program) and the desired resource.

If the TIP access control function were actively enforced, then to use a Host from a TIP, the TIP user would be required to first authenticate himself to the TIP, next to open a logical connection to the service Host, and finally to authenticate himself to the Host before actually beginning to make use of the Host's services. Although the actual time and effort required of the user to complete these steps would not be large, many users, when faced with the possibility of TIP access control, have had strongly negative reactions to this process of "double login". Rather than perceiving the two instances of authentication as providing additional security, many users perceive the process as forcing them to do the "same thing" twice. To cure this perceived problem, modifications to the TIP and the TIPSER-RSEXEC would be required to make it possible for service Hosts to learn the identity of a TIP user based on the authentication data provided at the time of TIP login. This mechanism could be provided in such a way that only those Hosts choosing to make use of it would be required to modify their software, and only users

choosing to make use of it would lose the extra security barrier.

Once the TIP user is connected to the TIPSER-RSEXEC, it would be convenient if the user could choose a service Host and have the TIPSER-RSEXEC reconnect him to that Host without the user having to explicitly break his connection to the TIPSER-RSEXEC and then explicitly open a connection to the service Host. Ideally, the user would request not a particular service Host, but a particular service; and the TIPSER-RSEXEC would reconnect him to the site providing the desired service in the most responsive way or the most economical way or the way having some other desirable attribute. Finally, when finished with a service (or service Host), the user could be reconnected back to an available TIPSER-RSEXEC. All of this reconnection back and forth should be transparent to the user, thus truly providing the appearance of a common (albeit virtual) executive.

Once such a virtual executive is conveniently available to TIP users, it becomes possible to think of additional features that can be added. For instance, the TENEX RSEXEC makes available to TENEX users a virtual file system which spans machine boundaries. It is a simple technical step to provide the TIP users (who, unlike TENEX users, have never had a file system) with a virtual file system. Another example: while the TIPSER-

RSEXEC has the capability (currently disabled) to permit users to leave messages for other users, it does not provide the capability for TIP users to receive such messages. Yet, through the concept of resource sharing, the potential capability to provide virtual mailboxes through which users can receive messages exists. Furthermore, through the redundancy inherent in the system, the virtual mailboxes could be provided in a way which would insure that a user's mail was accessible no matter which individual computers were down. A final example: once the TIP user is connected to the TIPSER-RSEXEC and is ready to use the services of some Host, and once it is possible for the user to call for service independent of Host, there is no need to retain in the user's view the concept of the Host(s) from which service is obtained; rather, the virtual executive could be expanded to provide the virtual operating system from which all service is obtained.

To the extent that the virtual executive, the virtual mail service, the virtual operating system, and the like are made available to users, two changes in traditional computer operations are in order. First, the problem of unique user names arises. Traditionally, a user name had only to be unique to each local computer system. However, if users of many systems are to communicate through a single virtual mail system, keep their

files in a single virtual file system, be authenticated by a single virtual authentication system, and so on, then there is a clear need for universal user names. Our system provides for such universal names by allowing the use of a person's full name (i.e. first, middle, and last), along with the person's affiliation, although only the minimum data required for unique recognition is required.

The second necessary break with traditional computer operational practice is in the area of accounting and billing. Traditionally, each user makes arrangements with each center of computer service to which he desires access. With an integrated resource sharing system in which the existence of the individual Hosts is of minimal importance, it is highly desirable to have a system-wide accounting and billing system. The user should not have to execute a large number of contracts with individual sites or receive a large number of bills for computer service each month, especially when his use of these individual systems was not apparent to him. Rather, the user will want to execute one contract for all his computer service, or at most one contract for each type of system he desires, independent of the sites from which the service is obtained. Our system contains prototype mechanisms to facilitate such global accounting practices (in particular, for invoicing a TIP user for all his TIP use in a

month independent of the number of TIPs from which he received his TIP use).

It is interesting to note that the TIPSER-RSEEXEC system need not be necessarily limited to TIP use. Any Host needing similar functions out of a desire for standardization or because the Host is unable or unwilling to provide the services itself could make use of the TIPSER-RSEEXEC. In general, we believe that terminal concentrator Hosts such as the TIP should make use of the TIPSER-RSEEXEC, as we assert it is the proper function of such terminal concentrators to specialize in the handling of terminal I/O and to leave other functions to other Hosts. We assert that the reverse is also true. Service Hosts should generally specialize in the handling of application functions and leave the details of terminal I/O to a terminal concentrator. We believe our system properly supports such specialization of function, and that is it economically advantageous to make use of such a system whenever possible.

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, Ma. 02138		2a. REPORT SECURITY CLASSIFICATION Unclassified	
2b. GROUP			
3. REPORT TITLE QUARTERLY TECHNICAL REPORT NO. 2 INTERFACE MESSAGE PROCESSORS			
4. DESCRIPTIVE NOTES (Type of report and, inclusive dates) 1 April 1975 to 30 June 1975			
5. AUTHOR(S) (First name, middle initial, last name) Bolt Beranek and Newman Inc.			
6. REPORT DATE July 1975		7a. TOTAL NO. OF PAGES 80	7b. NO. OF REFS
8a. CONTRACT OR GRANT NO. F08606-75-C-0032		9a. ORIGINATOR'S REPORT NUMBER(S) Report No. 3106	
b. PROJECT NO.		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
c.			
d.			
10. DISTRIBUTION STATEMENT Distribution unlimited			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY	
13. ABSTRACT The ARPA computer network is a packet-switching store-and-forward communications system designed for use by computers and computer terminals. After a brief overview of various aspects of network operations and network-related developments, network performance is studied in detail including discussion of a number of software changes aimed at performance improvement. Some future network developments are discussed in brief. A final installment on the subject of TIP access control and accounting is included.			

DD FORM 1473

(PAGE 1)

5/9 0101-801-0811

UNCLASSIFIED

Security Classification

A 1140-

UNCLASSIFIED

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Computers and Communication						
Store and Forward Communication						
ARPA Computer Network						
Packets						
Packet-switching						
Interface Message Processor						
IMP						
Terminal IMP						
TIP						
Pluribus						
Satellite IMP						
Access Control						
Accounting						
Private Line Interface						
PLI						
Broadcast Communications						
Acknowledgment						
Retransmission						

DD FORM 1 NOV 65 1473 (BACK)

S/N 0101-907-6921

UNCLASSIFIED

Security Classification

A-31403